DOI URL:https://doi.org/10.63944/krg.NHE2-1
Vol.2 No.1

Development of a Practical Plan to Improve Basic Programming Skills for Third-Grade Primary School Students: A Case Study of Xincheng Primary School in Lingui District

Zhaohong Yang, Nipaporn Khamcharoen*

Faculty of Education, Dhonburi Rajabhat University

Bangkok, Thailand | 276536721@qq.com

***Corresponding Author: Nipaporn Khamcharoen, Faculty of Education, Dhonburi Rajabhat University Bangkok, Thailand | nipaporn.k@dru.ac.th**

Abstract: With the global emphasis on programming education, cultivating primary school students' basic programming skills has become a key component of quality-oriented education. This study aims to investigate the opinions of third-grade students at Xincheng Primary School in Lingui District on improving basic programming skills and develop targeted academic administration practical plans. A mixed-method research design was adopted, including a questionnaire survey of 184 third-grade students and focus group discussions with 5 experts. The questionnaire, validated by experts (IOC: 0.67-1.00) with high reliability (Cronbach's α=0.895), was used to collect data on students' general information and their perceptions of programming learning. Quantitative data analysis employed percentage, mean, and standard deviation, while qualitative analysis of expert opinions informed the development of practical plans. The findings revealed that students' overall perception of programming skills was at a high level ($\bar{x}$=3.93, S.D.=1.05), with basic computer operation skills scoring the highest ($\bar{x}$=4.00, S.D.=1.13) and basic programming concepts scoring the lowest ($\bar{x}$=3.88, S.D.=1.05). Key challenges included standardized file management, understanding abstract concepts, balancing technical effects with project themes, flowchart application, and collaborative documentation. Based on these results, practical plans were developed covering five dimensions: basic computer operation skills, basic programming concepts, graphical programming skills, logical thinking and problem-solving ability, and project practical skills. This study provides practical references for optimizing primary school programming education and enriches the theoretical system of information technology education management. The developed practical plans offer actionable strategies for teachers to address students' specific needs and improve the quality of programming teaching.

Keywords: Basic Programming Skills; Third-Grade Primary School Students; Academic Administration Practical Plans; Programming Education

1. Introduction

The rapid development of information technology has elevated programming from a specialized skill to a basic literacy in modern society. Globally, developed countries such as the United States and the United Kingdom have long integrated programming education into their basic education frameworks. They adopt diverse teaching methods, including project-based learning and gamified learning, to foster students' computational thinking and innovative abilities. In China, the State Council's "Development Plan for the New Generation of Artificial Intelligence"(2017) explicitly requires the inclusion of artificial intelligence-related courses and the gradual promotion of programming education in primary and secondary schools. In recent years, various regions have launched pilot programs, which indicates a vigorous development trend of programming education.

However, the implementation of programming education in Chinese primary schools is still in the initial stage and faces multiple challenges. The programming curriculum system lacks completeness and coherence, so it cannot meet students' diverse needs. Teaching methods remain single; traditional lecture-based teaching dominates this field. This teaching method lacks interactivity and practicality, thus failing to stimulate students' learning enthusiasm. High-quality teaching resources, such as textbooks, online courses and practical platforms, are insufficient. This insufficiency restricts the improvement of teaching effectiveness. In addition, primary school information technology teachers often have deficiencies in programming teaching capabilities. For example, their professional knowledge is outdated, and their practical guidance skills are inadequate. These deficiencies hinder the improvement of programming education quality [1].

Third-grade primary school students are in a critical period of cognitive development. Their logical thinking abilities are improving rapidly, and they can understand complex concepts. This stage is ideal for introducing programming education, because it helps cultivate students' logical reasoning, problem-solving and innovative thinking skills. These skills are transferable across disciplines and essential for their future career development . However, third-grade students also face unique challenges in programming learning. They have difficulty in grasping abstract concepts, weak awareness of logical structure and insufficient practical application capabilities[2]. These challenges highlight the necessity of formulating targeted practical plans to improve their basic programming skills. Against this background, this study aims to investigate the opinions of third-grade students at Xincheng Primary School in Lingui District on improving basic programming skills, and develop a practical academic management plan tailored to enhance these skills for the target group.

This research has important theoretical and practical value. Theoretically, it enriches the theoretical system of teachers' professional development in information technology education by exploring the constituent elements and improvement paths of primary school students' basic programming skills. It also promotes interdisciplinary integration by connecting programming education with mathematics, science and art, and expands the application of educational theories in specific subject areas. Practically, the developed practical plan provides clear directions and operable methods for teachers at Xincheng Primary School. It helps teachers identify the strengths and weaknesses of students' learning and implement targeted teaching strategies. This not only improves teaching effectiveness and stimulates students' learning interest, but also provides a replicable reference for balancing the development of programming education in small-scale primary schools. Furthermore, by cultivating students' programming skills and computational thinking, this research

contributes to nurturing the basic technological literacy of young people and lays a foundation for the long-term development of China's information technology and artificial intelligence industries.

2. Research Methods

This study adopted a mixed-method research design, combining quantitative and qualitative approaches to comprehensively investigate students' programming learning status and develop a scientific Practical Plan, with Xincheng Primary School in Lingui District, Guilin City selected as a case study—the population included 340 third-grade students from the school who possess certain learning abilities, thinking skills, and the potential to master basic programming skills, and the sample size was determined using the Taro Yamane formula (Yamane, 1973)[3] with a confidence level of 95% (e=0.05), resulting in 184 students selected through stratified random sampling from 7 third-grade classes to ensure representation across different classes. The research instrument was a self-developed questionnaire consisting of two parts, namely general information (including gender, class, programming learning duration, participation in extracurricular programming classes, and available learning equipment) and opinions on programming skills improvement (covering five dimensions—basic computer operation skills, basic programming concepts, graphical programming skills, logical thinking and problem-solving ability, and project practical skills—with a total of 25 items), which was validated by 3 experts using the Index of Consistency (IOC) with each item's IOC value ranging from 0.67 to 1.00 (indicating good content validity) and yielded a Cronbach's Alpha coefficient of 0.895 (greater than 0.8, demonstrating high internal consistency and reliability). For data collection, the researcher obtained permission from the school principal in advance, explained the purpose and confidentiality of the survey to the students who completed the questionnaire voluntarily via online email, resulting in 184 valid questionnaires with an effective recovery rate of 100%, while five experts in the fields of primary school information technology education, programming teaching, and curriculum development were invited to participate in focus group discussions focusing on analyzing students'programming learning challenges and formulating the Practical Plan, with their opinions recorded and analyzed to revise and improve the draft. For data analysis, the collected questionnaire data were organized and analyzed using descriptive statistics including frequency, percentage, mean, and standard deviation, with the five-point Likert scale (4.51-5.00 as highest, 3.51-4.50 as high, 2.51-3.50 as middle, 1.51-2.50 as low, and 1.00-1.50 as lowest) [4] used to interpret the results, and the data from expert focus group discussions were analyzed using content analysis to extract key opinions and suggestions for validating and refining the Practical Plan, ensuring its scientificity, feasibility, and applicability.

Table 1. Reliability Analysis of the Questionnaire

Scale	Number of Items	Cronbach's Alpha
Student Questionnaire	25	0.895

3. Research Results

3.1. General Information of Students

The general information of the 184 surveyed students is presented in Table 2. In terms of gender distribution, males accounted for 55.43% (102 students) and females for 44.57% (82 students), showing a slight male predominance. Regarding class distribution, the sample

covered 7 third-grade classes, with Class 3(2), Class 3(3), and Class 3(7) each accounting for 15.22%, followed by Class 3(4) (14.67%), Class 3(1) and Class 3(5) (13.59% each), and Class 3(6) (12.50%), indicating a relatively balanced distribution across classes.

In terms of programming learning duration, 34.78% (64 students) had studied programming for 3-6 months, 24.46% (45 students) for 7-12 months, 16.85% (31 students) for more than 1 year, 18.48% (34 students) for less than 3 months, and only 5.43% (10 students) had no programming learning experience. This indicates that approximately three-quarters of the students had more than 3 months of programming learning experience, possessing a certain foundation.

Regarding participation in extracurricular programming classes, 88.59% (163 students) had participated in such classes, while 11.41% (21 students) had not, reflecting the popularity of programming learning in extracurricular activities. In terms of available learning equipment, 46.20% (85 students) had a computer or tablet at home, and 53.80% (99 students) used a mobile phone as their sole learning device, with no students lacking dedicated learning equipment.

Table 2. General Information of Students (n=184)

Category	Subcategory	Frequency	Percentage (%)
Gender	Male	102	55.43
	Female	82	44.57
Class	Class 3(1)	25	13.59
	Class 3(2)	28	15.22
	Class 3(3)	28	15.22
	Class 3(4)	27	14.67
	Class 3(5)	25	13.59
	Class 3(6)	23	12.50
	Class 3(7)	28	15.22
Programming Learning Duration	Never learned	10	5.43
	Less than 3 months	34	18.48
	3-6 months	64	34.78
	7-12 months	45	24.46
	More than 1 year	31	16.85
Participation in Extracurricular Programming Classes	Yes	163	88.59
	No	21	11.41
Home Learning Equipment	Computer/Tablet	85	46.20
	Only mobile phone	99	53.80
	No dedicated equipment	0	0.00

3.2. Analysis of Primary School Information Third Grade Students in Learning Programming Skills

The students' opinions on programming skills learning across the five dimensions are presented in Table 3. The overall mean score was 3.93 (S.D.=1.05), falling into the "high"level, indicating that students generally held positive perceptions of their programming learning status.

Among the five dimensions, basic computer operation skills scored the highest ($\bar{x}$=4.00, S.D.=1.13), followed by project practical skills ($\bar{x}$=3.94, S.D.=1.11), logical thinking and problem-solving ability ($\bar{x}$=3.93, S.D.=1.07), graphical programming skills ($\bar{x}$=3.90, S.D.=1.02), and basic programming concepts ($\bar{x}$=3.88, S.D.=1.05).

In the basic computer operation skills dimension, the highest-scoring items were "independently turning on/off the computer and safely using the mouse and keyboard" ($\bar{x}$=4.07, S.D.=0.99), "connecting headphones to the computer and adjusting the volume" ($\bar{x}$=4.04, S.D.=1.09), and "saving Scratch projects with clear filenames" ($\bar{x}$=3.96, S.D.=1.09), indicating that students had a solid grasp of basic computer operations.

In the project practical skills dimension, the top three items were "presenting and explaining design ideas to the whole class" ($\bar{x}$=3.97, S.D.=1.07), "adding instructions to help others use the program" ($\bar{x}$=3.96, S.D.=1.12), and "listening to classmates' ideas before deciding in group work" ($\bar{x}$=3.94, S.D.=1.06), reflecting students' strong communication, collaboration, and user awareness.

In the logical thinking and problem-solving ability dimension, the highest-scoring items were "drawing simple flowcharts to plan programs" ($\bar{x}$=4.04, S.D.=1.07), "breaking down big problems into small steps to solve" ($\bar{x}$=3.97, S.D.=1.10), and "designing simple tasks" ($\bar{x}$=3.89, S.D.=1.08), demonstrating students' good logical thinking and planning abilities.

In the graphical programming skills dimension, the top three items were "making a character say something using the 'say' block" ($\bar{x}$=3.98, S.D.=1.02), "adding a new background from the library" ($\bar{x}$=3.91, S.D.=1.04), and "adjusting the shape, size, and color of the character" ($\bar{x}$=3.90, S.D.=1.04), indicating students' proficiency in using Scratch for interactive and creative operations.

In the basic programming concepts dimension, the highest-scoring items were "understanding the role of 'instructions'" ($\bar{x}$=3.93, S.D.=1.00), "understanding the meaning of 'conditional judgment'"($\bar{x}$=3.89, S.D.=1.05), and "knowing how to use the 'stop' block to end a program" ($\bar{x}$=3.89, S.D.=1.03), showing that students had a basic grasp of core programming concepts, though there was room for improvement in understanding abstract concepts.

Table 3. Students' Opinions on Programming Skills Learning (n=184)

Questions	n=184		Level of Opinions
	$\bar{X}$	S.D.	
Basic computer operation skills			
1. You can turn on and off the computer independently and use the mouse and keyboard safely.	4.07	0.99	High
2. You can create folders to save programming works and find files quickly.	4.01	0.99	High
Table 4.2 Opinion of primary school information third grade students in learning programming skills (Cont.)			
Questions	n=184		Level of Opinions
	$\bar{X}$	S.D.	
3. You can skillfully open the Scratch software and use the "green flag" to run the program.	3.92	1.04	High
4. You can connect headphones to the computer and adjust the volume.	4.04	1.09	High
5. You know how to save my Scratch project with a clear filename (e.g., "My Game").	3.96	1.09	High
Total	4.00	1.13	High
The basic concepts of programming			
6. You understand the role of "instructions" (for example: let the character move 10 steps).	3.93	1.00	High
7. You know that "loops" can make the code repeat (for example: let the	3.85	1.10	High

character turn 5 times).			
8. You understand the meaning of "conditional judgement" (for example: if it hits the edge, it will bounce back).	3.89	1.05	High
9. You understand that changing numbers in commands affects results (e.g., changing "move 10 steps" to "move 20 steps").	3.85	1.06	High
10. You know how to use the "stop" block to end a program.	3.89	1.03	High
Total	3.88	1.05	High

Table 4.2 Opinion of primary school information third grade students in learning programming skills (Cont.)

Questions	n=184 $\bar{X}$	S.D.	Level of Opinions
Graphical programming skills			
11. You can use Scratch building blocks to combine simple animations (such as character walking).	3.84	1.09	High
12. You can adjust the shape, size and color of the character.	3.90	1.04	High
13. You can add background music or sound effects to the program.	3.89	1.11	High
14. You can make a character say something by using the "say" block.	3.98	1.02	High
15. You can add a new background to my project from the library.	3.91	1.04	High
Total	3.90	1.02	High
Logical thinking and problem-solving ability			
16. When you encounter problems in programming, you will break down big problems into small steps to solve them.	3.97	1.10	High
17. When the program fails, you will check whether the order of the building blocks is reasonable.	3.87	0.99	High
18. You can design a simple task (such as letting the character avoid obstacles and reach the end).	3.89	1.08	High
19. You can draw a simple flowchart to plan my program (e.g., "start → move → jump → end").	4.04	1.07	High

Table 4.2 Opinion of primary school information third grade students in learning programming skills (Cont.)

Questions	n=184 $\bar{X}$	S.D.	Level of Opinions
Total	3.93	1.07	High
20. When a character moves incorrectly, you try different blocks to fix it.	3.89	1.08	High
Project practical skills			
21. You can complete a programming work independently (such as a self-introduction animation).	3.92	1.01	High
22. When working in a group, you can divide the work with my classmates to complete the task.	3.92	1.02	High
23. After completing the work, you can show and explain the design ideas to the whole class.	3.97	1.07	High
24. You can add instructions to help others use my program (e.g., "Press space to start").	3.96	1.12	High
25.When working in a group, you listen to classmates' ideas before deciding.	3.94	1.06	High
Total	3.94	1.11	High
Total	3.93	1.05	High

3.3. Practical Plan for Improving the Basic Programming Skills of Third-Grade Primary School Students at Xincheng Primary School in Lingui District

Based on the questionnaire results and expert focus group discussions, Practical plans for improving third-grade students' basic programming skills were developed, covering five core dimensions with specific strategies (Table 4).

Table 4. Implementation Strategies

Core Skill Dimensions	Specific Implementation Strategies
Basic Computer Operation Skills	1. Prioritize standardized file saving habits in the early stages of programming instruction, promoting the naming convention of "Name + Date + Project Name" through demonstrations and repeated practice. 2. Enhance student engagement through gamified activities such as "naming challenges" and "file scavenger hunts" to reinforce standardized file management. 3. Optimize the storage interface design of the teaching platform to make save paths and file naming more intuitive and user-friendly for younger students.

	4. Incorporate a "save confirmation + naming review" process after each programming task to help students internalize the complete operation chain of "finish the work - save it - find it".
Basic Programming Concepts	1. Adopt a "visualization + comparative experiment" approach to teach abstract concepts such as parameter modification, guiding students to establish logical connections between parameters and results. 2. Encourage students to experiment with different parameter values and observe changes, fostering a trial-and-error mindset and parameter sensitivity. 3. Use real-life analogies (e.g., comparing parameters to seasoning a dish) and interactive tools (e.g., sliders on interactive whiteboards) to visualize abstract concepts. 4. Design contextualized exercises for loops and conditional judgments to strengthen students' application capabilities in diverse scenarios.
Graphical Programming Skills	1. Adhere to the teaching principle of "theme first, material selection later", guiding students to clarify creative themes through storyboards or verbal descriptions before selecting materials. 2. Set appropriate restrictions on the number of characters, backgrounds, or special effects in tasks to avoid overemphasis on technical operations and neglect of project themes. 3. Implement a three-step creative process: "sketch conception - material planning - project construction" to help students organize their ideas and focus on content expression. 4.Emphasize creative integrity in project presentations and evaluations, guiding students to understand that technology serves expression.
Logical Thinking and Problem-Solving Ability	1. Design hierarchical task sequences aligned with students' cognitive levels, starting from familiar daily scenarios (e.g., breaking down the "toothbrushing process") and gradually transitioning to complex programming tasks. 2. Cultivate step decomposition skills through "level-breaking task cards" and role-playing activities (e.g., "robot instruction games"). 3. Simplify flowchart symbols and use "life task diagrams" (e.g., morning getting up process) to lower the understanding threshold, and display Scratch building blocks and flowchart structures in parallel to establish cognitive connections. 4. Introduce systematic debugging training, focusing on logical flow checks and error identification.
Project Practical Skills	1. Clarify role division in collaborative projects, assigning roles such as "programmer", "designer", and "recorder" to reinforce documentation responsibilities. 2. Provide structured document templates (e.g., "What I did - Why I did it") and allow a combination of oral explanations, audio transcriptions, and graphic materials to lower the writing barrier. 3. Establish peer review mechanisms (e.g., "manual scorecards") to encourage students to examine document clarity, completeness, and reproducibility. 4. Organize regular project sharing sessions to provide students with opportunities to present their works and explain design ideas, improving their expression and communication skills.

4. Discussion

4.1. Opinion of Primary School Information Third Grade Students in Learning Programming Skills

The study found that third-grade students at Xincheng Primary School have an overall high perception of their programming skills ($\bar{x}$=3.93, S.D.=1.05). This result reflects the students' positive learning attitudes and good learning outcomes. It is consistent with the findings of Sáez-López et al.[5], who noted that primary school students can effectively master basic programming skills through appropriate teaching methods.

The participation rate in extracurricular programming classes reaches 88.59%. This high rate indicates that students have strong extrinsic motivation and intrinsic acceptance of programming learning, and these factors may contribute to their positive self-perceptions of programming skills.

Among the five assessment dimensions, basic computer operation skills obtained the highest score ($\bar{x}$ =4.00, S.D.=1.13). This result aligns with Nagyová's [6] view that basic computer operations are core prerequisites for information literacy and programming learning. Students' proficiency in power-on/off procedures, input device usage, and file management provides a solid foundation for their subsequent programming activities. This proficiency also demonstrates the effectiveness of daily computer basics instruction.

Project practical skills ranked second ($\bar{x}$ =3.94, S.D.=1.11). This score indicates that students can effectively apply programming knowledge to real-world scenarios. Their strong performance in presenting design ideas, adding user instructions, and collaborating in groups reflects the value of project-based learning in programming education. This finding is

consistent with Resnick et al.'s [7] research, which showed that Scratch-based project practice fosters students' creative expression and collaborative skills. It also suggests that current programming instruction has achieved good results in cultivating students' practical application capabilities.

Logical thinking and problem-solving ability scored 3.93 (S.D.=1.07). Students demonstrated good planning skills (e.g., using flowcharts) and problem decomposition abilities. This result supports Erümit's [8] conclusion that programming education effectively promotes students' logical thinking and problem-solving skills. However, students performed relatively weakly in mastering abstract logical structures (e.g., standardized flowchart symbols). This weakness indicates a mismatch between the abstraction of symbolic language and students' cognitive development level, which is a common challenge for primary school students [5].

Graphical programming skills scored 3.90 (S.D.=1.02). Students are proficient in using Scratch for interactive and creative operations, such as designing character dialogues and modifying backgrounds. This observation is consistent with Sapounidis et al.'s [9] finding that graphical programming tools like Scratch are user-friendly for beginners, as they lower the entry barrier to programming. However, experts noted that students tend to focus on visual effects and neglect project themes. This phenomenon is similar to Simon et al.'s research result that students often become overly entangled in technical details when using graphical programming tools.

Basic programming concepts scored the lowest ($\bar{x}$=3.88, S.D.=1.05) among the five dimensions. This score indicates that students face challenges in understanding abstract concepts such as parameters and logical structures. This result is consistent with the cognitive characteristics of young students, who prioritize practical application over theoretical internalization [10]. Students' weak understanding of parameter modifications and conditional statements may hinder their ability to implement complex interactive logic. This highlights the need for optimized teaching strategies in this area.

The study also identified key challenges faced by students: insufficient standardized file management habits, difficulty in grasping abstract programming concepts, overemphasis on technical effects at the expense of project themes, weakness in the flexible application of flowcharts, and inadequate documentation skills in collaborative projects. These challenges are consistent with the findings of previous studies , and they emphasize the need for targeted interventions to address students' specific learning needs.

4.2. Practical plan for Improving the Basic Programming Skills of Third-Grade Primary School Students at Xincheng Primary School in Lingui District

The developed practical plans integrate students' learning needs with expert opinions and cover five core dimensions of basic programming skills. These plans conform to relevant educational theories and research findings, thus demonstrating scientificity and feasibility.

Regarding basic computer operation skills, the practical plans emphasize standardized file management and interface optimization. This emphasis aligns with Saçkes et al.'s view that sufficient practice and user-friendly tools can improve students' computer operation skills. Meanwhile, the gamified activities proposed in the plans can enhance student engagement and reinforce good learning habits. This design is supported by Prykhodchenko et al.'s finding that gamification effectively promotes programming learning.

For basic programming concepts, the plans adopt a "visualization + comparative experiment" approach, which addresses the difficulty in teaching abstract programming concepts. This approach is consistent with Sáez-López et al.'s recommendation that

educators should use visualization tools and real-life analogies in programming instruction. Additionally, the plans emphasize trial-and-error practice and contextualized application. This emphasis helps students establish logical connections between programming concepts and practical applications, thereby fostering deep understanding instead of mechanical memorization.

In terms of graphical programming skills, the practical plans advocate a “theme first, material selection later” principle, which solves the problem that students often neglect project themes during learning. This principle aligns with Kaučič and Asič's view that Scratch should be used to foster students’ creativity and content expression rather than merely focus on technical operations. Moreover, the plans propose a three-step creative process, which provides students with a structured framework to organize their ideas and improves the coherence and integrity of their programming projects.

Regarding logical thinking and problem-solving abilities, the plans adopt hierarchical task design and simplified flowcharts. This design is consistent with Shin and Park's [11] finding that tiered tasks can effectively improve students’ problem-solving abilities. Furthermore, the plans use life task diagrams and role-playing activities, which reduce students' cognitive load and make abstract logical structures more understandable. This practice aligns with Milková and Hulková’s [12] emphasis on developing algorithmic thinking through practical activities.

For project practical skills, the plans implement role division and structured templates, which address the weakness of students’ documentation skills. This design is supported by Hiltunen's [13] view that programming learning should emphasize social interaction and collaboration. Additionally, the peer review mechanism and project sharing sessions in the plans enhance students’ communication and expression skills, and foster their sense of responsibility and user awareness.

Overall, the practical plans are tailored to the cognitive characteristics and learning needs of third-grade students, integrating theoretical insights and practical experience. They provide teachers with a systematic and actionable framework to improve the quality of programming teaching and address the key challenges identified in the study. Meanwhile, the plans emphasize the balance between technical skill training and creative expression, which aligns with the goal of cultivating comprehensive and innovative talents in modern education.

5. Recommendations for Further Study

5.1. Recommendations Based on Research Findings

In terms of the cultivation of students’ basic computer operation skills, schools should develop a “Third Grade Programming Operation Manual” that contains illustrated instructions on file saving, naming conventions, and common hardware operations. Meanwhile, schools ought to organize quarterly “Operational Skills Competitions”, such as competitions for file search speed and standardized naming, to reinforce students’ good operation habits through gamification. Additionally, teachers should optimize the school's programming teaching platform by setting default naming templates and adding a one-click save function.

In the teaching of basic programming concepts, educators should compile a “Collection of Real-Life Programming Concept Case Studies” to help students understand abstract programming concepts through familiar daily scenarios. Teachers need to design a “Parameter Exploration Worksheet” for students to record their observations of the changes in different parameter values. In classroom teaching, teachers should use interactive whiteboards and slider tools to visualize the changes of parameters in real time.

Regarding the improvement of students' graphical programming skills, teachers should implement the "Three-Step Creative Process" (including concept sketching, material planning, and project development) with the help of paper templates. Schools should set up "themed project tasks", such as "Environmental Protection Animation", to guide students to focus on the presentation of project content. Besides, schools ought to establish a themed "material resource library" to facilitate students' efficient search for relevant materials.

In the process of cultivating students' logical thinking and problem-solving abilities, teachers should design step-by-step task cards, such as "Level 1: Imitate Steps", "Level 2: Complete Steps", and "Level 3: Independently Design Steps", to meet the learning needs of students with different ability levels. Educators should create "life process flowcharts" as introductory materials for students to learn programming flowcharts. Before students carry out digital programming operations, teachers can use physical building blocks to help them simulate Scratch logic.

In terms of the cultivation of students' project practical skills, schools should create "Collaborative Project Role Cards" to clarify the responsibilities of each student in collaborative projects. Teachers need to develop "Project Document Templates" with fill-in-the-blank instructions to reduce the difficulty for students to complete project documents. Moreover, schools should organize regular "project sharing sessions" to improve students' expression skills and enhance their learning confidence.

5.2. Recommendations for Further Study

This study has several limitations, and these limitations provide clear directions for future research. First, the sample of this study was limited to 184 third-grade students from a single primary school, and this sampling limitation may affect the generalizability of the research findings. Therefore, future research should expand the sample size, include multiple primary schools in both urban and rural areas, explore regional differences in programming education, and verify the applicability of the proposed practical plans.

Second, this study primarily focused on students' perspectives and failed to fully investigate the impact of teacher-related factors, including teachers' programming teaching capabilities and training requirements. Future research could explore the relationship between teacher-related factors and students' programming skill development, and further develop corresponding support strategies for teachers.

Third, the long-term effectiveness of the developed practical plans in programming education remains to be verified. Future research could conduct long-term follow-up studies to observe changes in students' programming skills over an extended period, evaluate the sustainability of the practical plans, and make necessary revisions to these plans based on their practical application effects.

Fourth, although this study adopted a mixed-method research design, it did not deeply explore the causal relationships among the key variables involved. Future research could employ more advanced statistical methods, such as structural equation modeling, to analyze the causal relationships between teaching strategies, students' learning behaviors, and their programming skill development outcomes.

Finally, with the continuous advancement of educational technology, new programming tools and teaching methods (e.g., artificial intelligence-assisted teaching) are constantly emerging. Future research could explore the application of these new technological tools and methods in primary school programming education, and update the existing practical plans accordingly to keep up with the trends of educational innovation.

References

[1] Kalelioglu, F., & Gülbahar, Y. (2014). The effects of teaching programming via Scratch on problem solving skills: A discussion from learners' perspective. Informatics in Education, 13(1), 33-50.

[2] Tsarava, Z., Jongsma, K. R., & van der Maas, H. L. (2022). The development of computational thinking in primary school: A cross-sectional study. Journal of Educational Psychology, 114(3), 104425.

[3] Yamane, T. (1973). Statistics: An introductory analysis (3rd ed.). Harper and Row.

[4] Wongwanich, S. (2005). Basic research methodology. Bangkok: Chulalongkorn University Press.

[5] Sáez-López, J. M., Marcos Román-González, & Esteban Vázquez-Cano. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. Computers & Education, 97, 129-141.

[6] Nagyová, I. (2016). Constructivism in teaching of basic computer skills. Procedia - Social and Behavioral Sciences, 234, 120-125.

[7] Resnick, M., Maloney, J., Monroy-Hernández, A., et al. (2009). Scratch: Programming for all. Communications of the ACM, 52(11), 60-67.

[8] Erümit, E. (2020). The effect of scratch programming on the problem-solving skills of primary school students. Education and Information Technologies, 25(2), 1013-1037.

[9] Sapounidis, T., Demetriadis, S., & Stamelos, I. (2019). Evaluating the usability of scratch for teaching programming to primary school students. Education and Information Technologies, 24(1), 67-78.

[10] Hijón-Neira, R., Gómez-Sánchez, E., & García-Peñalvo, F. J. (2020). Metaphors and visualization tools to teach programming concepts to primary school students. Computers & Education, 155, 217800-217815.

[11] Shin, S., & Park, P. (2014). A study on the effect affecting problem solving ability of primary students through the scratch programming. Advanced Science and Technology Letters, 59(1), 117-120.

[12] Milková, E., & Hulková, A. (2013). Algorithmic and logical thinking development: Base of programming skills. WSEAS Transactions on Computers, 12(2), 41-51.

[13] Hiltunen, M. (2016). Social constructivist approaches to teaching programming in primary school (Master's thesis). University of Jyväskylä.